A Reward Analysis of Reinforcement Learning from Large Language Model Feedback

Anonymous authors
Paper under double-blind review

Abstract

The correct specification of reward models is a well-known challenge in Reinforcement Learning. Hand-crafted reward functions often lead to inefficient or suboptimal policies and may not be aligned with user values. Reinforcement Learning from Human Feedback (RLHF) is an important technique that can mitigate such issues, however, the collection of human feedback can be laborious and biased. Recent works have solicited feedback from value-aligned large language models rather than humans to reduce or eliminate human effort, however, there is no clear consensus on how to best apply large models to this task. In this paper, we study the advantages and limitations of Reinforcement Learning from Large Language Model Feedback (RL-LLM-F) and propose a simple, effective method for soliciting and applying feedback as a potential-based shaping function. We empirically examine the influence of various factors on both the learned reward model and the resulting policy, and explore several techniques for balancing the accuracy and sparsity of a reward model. Lastly, we show that treating LLM-based feedback as a potential-based scoring function eliminates the need for complex post-processing and leads to improved convergence speed and policy return.

1 Introduction

The correct specification of task rewards is a well-known challenge in Reinforcement Learning (RL) (Leike et al., 2018). Complex tasks often necessitate complex reward models, particularly as shaping terms may be required to guide exploration. However, hand-crafting complex reward functions often lead to a phenomenon known as *reward hacking*, wherein an agent learns to repeatedly exploit a reward function for increased returns while yielding unexpected or undesired behavior (Skalse et al., 2022). Reward hacking is symptomatic of the broader challenge of *value alignment*, in which it is difficult for a human domain expert to fully and accurately specify the desired behavior of the learned policy in terms of a reward function.

Reinforcement Learning from Human Feedback (RLHF) has demonstrated considerable effectiveness at tackling the value-alignment problem (Christiano et al., 2017). Rather than directly crafting reward functions, model outputs are ranked by humans according to their internal values. These preference rankings are then used to learn a reward function which helps guide model training and facilitate value alignment. This process is extremely costly in terms of human efforts, however, requiring scores of human laborers (El Asri et al., 2016).

More recently, pre-trained and value-aligned large language models have been used as a source of feedback in Reinforcement Learning in lieu of human annotaters (Lee et al., 2023). However, LLMs are well known to hallucinate and present false information as fact (Zhang et al., 2023), affecting the accuracy and reliability of the resulting rankings. As a result, many existing methods employ task-specific post-processing techniques over learned reward models to make them effective, e.g. filtering (Klissarov et al., 2023), and scaling (Christiano et al., 2017). In this work, we explore the limitations of RL from Large Language Model Feedback (RL-LLM-F) through the following

questions: 1) how do errors in LLM-based rankings affect the resulting policy performance; and 2) what can we do to mitigate the effect of such errors, e.g. issuing *fewer* but *more accurate* rankings. Based on our findings, we propose a straightforward and effective approach for RL-LLM-F which learns a potential-based scoring function from LLM-based preference rankings which is then converted to a dense reward function based on score differences. This approach obviates the need for tedious, task-specific post-processing and we empirically show that it improves convergence speed and policy returns in complex grid-world environments where standard reward functions fail.

2 Related Works

Constructing rewards based on human feedback has a long history (Thomaz et al., 2006). To efficiently use human domain knowledge and provide more generalizable rewards, human preference on episode segments (Sadigh et al., 2017; Christiano et al., 2017; Bıyık et al., 2019) and human demonstrations (Bıyık et al., 2022) are distilled into models which serve as reward functions for RL. The method has witnessed great success in complex domains where rewards are difficult to specify such as training large language models (LLM) to align with human logic and common sense (Ziegler et al., 2019; Ouyang et al., 2022).

One drawback of RLHF is still that its requirement of exhaustive human participation to provide demonstrations and feedback. LLM has shown deductive logic ability comparable to humans in recent years (Du et al., 2023), and is able to substitute humans in reward issuing (Kwon et al., 2023; Yu et al., 2023; Lee et al., 2023; Xie et al., 2023), or data collection and labeling for reward model training (Lee et al., 2023; Klissarov et al., 2023). While the former suffers from time and resource costs for training RL agents, the latter is becoming promising for training complex RL tasks (Wang et al., 2024).

Our work focuses on two main challenges of RLHF. One is the method of sampling states for annotation (Casper et al., 2023). Many RLHF works follow an on-policy style (Christiano et al., 2017; Lee et al., 2023; Wang et al., 2024), where the reward model and the agent's policy are alternatively updated, which is inefficient for the RL process. MOTIF (Klissarov et al., 2023) uses LLM to annotate single states with positive, negative, or neutral feedback, and then collects these states to train reward models. This simple pipeline decouples reward training with policy training, significantly accelerating the RL process. Nevertheless, evaluating a single state or comparing two arbitrary states is hard in complex tasks, in which ranking pairs of sequential states used by our work would be more general and practical.

The other challenge is the reward formulation. Many works train a model distilling LLM or human preference and use it as the reward model (Christiano et al., 2017; Wang et al., 2024; Klissarov et al., 2023; Lee et al., 2023), but this practice needs post-processing on outputs of the reward model such as filtering (Klissarov et al., 2023), and normalization (Christiano et al., 2017). Our work aims for a reward function trained with LLM without complex post-processing and environment rewards would be more general and adaptable to various practical scenarios.

It is also noticeable that the RLHF training process can be influenced by many factors. Casper et al. (2023), a survey on potential limitations and challenges of RLHF, points out that RLHF can be influenced by the feedback quality. Different LLMs have distinct probabilities of giving wrong feedback, thus leading to rewards of diverse quality. Casper et al. (2023) also suggests that comparison-based feedback may not be efficient and adequate to train reward models. Given the above challenges, the analysis of the influence of different LLM models, reward scales, and reward densities on policy training would provide valuable insights into reward shaping by RLHF.

3 Background

Reinforcement Learning: The basic Reinforcement Learning framework is that an agent interacts with an environment and receives a reward for its current action at each time step, learning an

optimal action policy to maximize the rewards over time. This procedure can be formulated as an infinite horizon discounted Markov Decision Process (MDP) (Sutton & Barto, 2018). At each discrete timestep t in this process, the agent observes environment state s_t and takes action a_t, leading to the next environment state s_{t+1} and a reward r_t. An MDP is represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is a reward function, $\mathcal{T}(s, a, s') = P(s'|s, a)$ is a transition function, and γ is a discount factor. A stochastic policy $\pi(a|s) : \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ indicates the probability that the agent selects action a given the state s. The agent's goal is to learn π maximizing the expected discounted return through training, given an initial state distribution.

Preference-based Reinforcement Learning: Our work is based on the framework of Preference-based Reinforcement Learning, where the reward function is learned from preference labels on agent behaviors (Christiano et al., 2017; Ibarz et al., 2018; Lee et al., 2021a;b). For a pair of states (s_a, s_b), an annotator gives a preference label y that indicates which state is ranked higher, considering which state is closer to the given task goal. The preference label $y \in \{0, 1\}$, where 0 indicates s_a is ranked higher than s_b, and 1 indicates s_b is ranked higher than s_a. Given a parameterized state-score function σ_ψ, which is commonly called the potential function and usually equated with the reward model r_ψ, we compute the preference probability of a state pair with the standard Bradley-Terry model (Bradley & Terry, 1952),

$$P_\psi[s_b \succ s_a] = \frac{\exp\left(\sigma_\psi(s_b)\right)}{\exp\left(\sigma_\psi(s_a)\right) + \exp\left(\sigma_\psi(s_b)\right)}, \tag{1}$$

where $s_b \succ s_a$ indicates s_b is ranked higher than the state s_a. With a preference dataset $D = (s_a^i, s_b^i, y^i)$, preference-based RL learns the state-score model σ_ψ by minimizing the cross-entropy loss, which aims to maximize the score difference between the state ranked higher and the state ranked lower:

$$\mathcal{L} = -\mathbb{E}_{(s_a, s_b, y) \sim \mathcal{D}} \left[\mathbb{I}\{y = (s_a \succ s_b)\} \log P_\psi[s_a \succ s_b] + \mathbb{I}\{y = (s_b \succ s_a)\} \log P_\psi[s_b \succ s_a] \right]. \tag{2}$$

The traditional RLHF approach falls into this category if the annotator is human, and traditional LLM-based RLHF simply replaces human annotation with LLM contexts, and the state-score model is directly used as the reward model. Our work is built upon this, while improving the sampling efficiency and generalizability.

4 Method

The RL-LLM-F pipeline consists of two stages: training the state-score model by collecting preference rankings from the LLM, and issuing rewards from this model while training a policy via RL.

We further break this down into the following four steps: (1) Randomly sample pairs of sequential states; (2) Query an LLM to rank the states in each pair concerning a natural language task description, e.g. "Goal to the green goal". The prompt contains a language task description, environment description, and Chain-of-Thought ranking examples (Wei et al., 2022) as context to generate preference labels for states in each pair. (3) Train the state-score model σ_ψ with the loss function in Equation 2, maximizing the score difference between the state ranked higher and the state ranked lower; (4) Train an optimal policy with feedback from the state-score model.

Prior works treat the state-score function as a reward function and directly issue rewards from it. This often leads to training instability as the rewards may have significant differences in scale, e.g. a reward of 0 in one state and -50 in a previous state, which needs to be corrected for via post-processing. However, in our work we treat the state-score function as a *potential function*, and define the reward to be the difference between the scores of successive state pairs, eliminating the need for post-processing.

$$r(s_t, a, s_{t+1}) = \sigma_\psi(s_{t+1}) - \sigma_\psi(s_t) \tag{3}$$

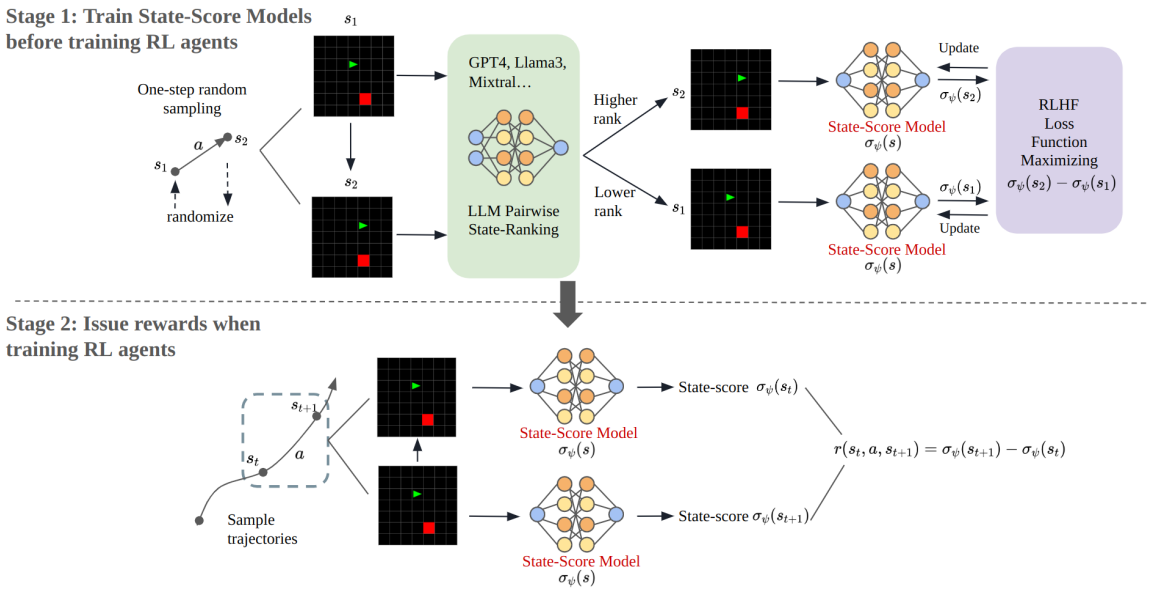

Figure 1: RL-LLM-F has two stages. In stage 1, a state-score model is trained with randomly sampled consecutive state pairs labeled with LLM preferences. For every pair, the first state is uniformly initialized from the entire state space to ensure a generalized state-score model with high sample efficiency. The second state is achieved from the first one by randomly executing an action. When training RL agents in stage 2, rewards are computed as state score differences between consecutive states in the sampled trajectory.

Moreover, the reward function in Equation 3 is naturally zero-mean, with positive rewards for actions progressing towards the given task goal and negative rewards for ineffective actions, and thus no post-processing is needed. The loss function of our approach in Equation 2 automatically optimizes the state-score difference, in order to help train our accurate and straightforward reward model. As RLHF utilizes helpful feedback, RL-LLM-F enables the agent to train better with rewards that are hard for RL agents with regular approaches.

Additionally, unlike many other LLM-based RLHF works (Christiano et al., 2017; Lee et al., 2023; Wang et al., 2024) which simply sample based on the agent's most updated policy and can have lots of inefficient ones when the agent is still exploring, in our work the LLM gives feedback by ranking pairs of sequential states. We sample these state pairs by randomizing the positions and status of all objects in the environment to get the first state, and then randomly choosing one action to get the second state. This method decouples the policy with data sampling for training the state-score model without leading to an improper distribution of the sampled state pairs. We empirically show the inefficient alternate training of policies and post-processing for reward scales can be reduced in our experiments.

5 Performance Analysis of RL-LLM-F

We evaluate RL-LLM-F performance characteristics over three Gridworld tasks according to the following factors: 1) Performance of distinct LLM models: use the state-score model ranking accuracy and RL agent training performance as evaluation criteria. 2) Reward scale analysis: the influence of multiple post-processing methods on reward magnitude. 3) Density and accuracy: while RL agents are in favor of denser rewards, LLM models are prone to make mistakes when handling subtle differences in state pairs. What is the trade-off? We also empirically analyzed the effectiveness of our reward function formulation based on state score differences.

5.1 Experiment Setup

We evaluate our methods in the grid-world environment, with 3 variations. The agent (solid triangle) needs to navigate to the target (solid rectangle). The doors are initially locked, and the agent must pick up the keys and use them to toggle the doors with corresponding colors. The layouts are shown in Figure 2. The baseline reward function is defined as 0 for failure, and $1 - 0.99n/n_{max}$ when the agent reaches the goal. n_{max} is the maximum time steps for each episode. n denotes the step count on success. We use PPO as the underlying policy-training framework (Schulman et al., 2017).

To perform this evaluation, we make the following assumptions: 1) There is one single agent in the globally observable grid world. 2) The agent does not have any knowledge about the task before training. 3) The reward model and LLM get access to the whole environment. 4) Given two states, LLM is capable of ranking which one is better for solving the various grid world tasks.

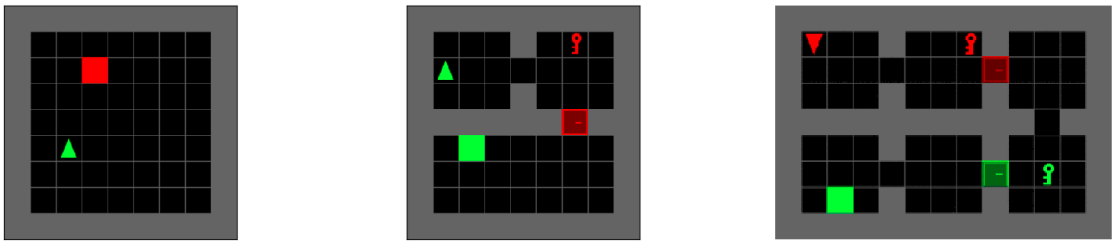

Figure 2: Grid-world environments 1 to 3 from left to right

5.2 Impacts of Distinct LLMs

We compare our variants that utilize 3 LLMs: Mixtral (Jiang et al., 2024), Llama3 (Touvron et al., 2023), and GPT 4 (Achiam et al., 2023). We also compare with the baseline reward introduced in the environment setup, which does not use LLM, and the RLHF approach where the rankings are done by humans and we consider as the ground-truth ranking.

We first examine how the LLMs can accurately rank states. The results are shown in Table 1. The sizes of the training datasets for the three environments are 2500, 3500, and 6000, whereas the sizes of the validation datasets are all 500. The RLHF approach ranks all state pairs using human inputs. Therefore, without the state-score model, its accuracy is 100.0%. Its state-score model we have trained for all environments has high accuracy values of 100.0%, 99.6%, and 99.8%. For LLM variants, if they output the same ranking as this ground-truth ranking, we consider them correct.

For all LLMs and all environments, the method that utilizes LLMs to train state-score models improves the ranking accuracy by simply asking LLMs to do so. When ranking without the state-score models, GPT 4 appears to be almost as good as the human ranking, and the other two LLMs are worse. However, the design of state-score models improves the ranking accuracy significantly of both these two worse LLMs. This is especially true for Llama3, as the state-score model trained with it reaches an accuracy much closer to GPT4 and ground-truth despite its poor accuracy without the state-score models, showing a high tolerance of our state-score model on LLM quality. GPT4 with a state-score model also has a slight improvement that enables it to be comparable to the ground-truth, and maintains to be the best among variants. This can be caused by some chain of logic encoded in the state-score model via minimizing its loss on the score difference of state pairs. The other possible reason is that when LLM answers the same query multiple times, it may make mistakes sometimes. Nevertheless, the correct ranks of this state pair dominate in the training dataset, and the state-score model can successfully extract the predominant correct information and exclude noisy data.

	RLHF Baseline	**Mixtral**	**Llama3**	**GPT4**
Env1(Feedback)	100.0%	75.9%	93.0%	100.0%
Env1(SSM)	100.0%	88.5%	98.0%	100.0%
Env2(Feedback)	100.0%	64.5%	89.4%	98.2%
Env2(SSM)	99.6%	74.0%	96.5%	98.4%
Env3(Feedback)	100.0%	60.4%	89.9%	98.6%
Env3(SSM)	99.8%	65.6%	95.7%	99.0%

Table 1: The Ranking Correctness Rates of Distinct Feedback and the Corresponding State-Score Models (SSM) in Three Envrionments

To examine how our variants behave and the effects of the ranking accuracy, we show the training curves for each of the environments in Figure 3. The performance is measured by the baseline reward of the environments discussed before. The RLHF serves as the upper bounds for our variants.

For the first environment, both the GPT-4 and the Llama3 variants enable the training to be as good as the ground-truth ranking. All of them enable the agent to converge faster compared to the baseline. However, the Mixtral variant could not help the agent to train to optimal, as the ranking accuracy is too low. This is similar to the second environment, where the poor Mixtral variant makes the training struggle. Notice that the baseline can train to the optimal when we use the same hyper-parameters in a related work, but converge much slower than our approaches.

The third environment is the hardest grid task, as the baseline could not train the agent to the optimal. Our GPT4 variant is slightly worse than RLHF in terms of convergence rate. However, the Llama3 variant appears to have a bigger variance. Poor ranking still influences the Mixtral variant, mis-directs the agent and thus it cannot learn the task at all. Overall we can see that ranking accuracy greatly affects our training method, and some of the LLMs may take the role of human ranking, even when they are not experts in its role. The state-score model may boost their ranking accuracy to still improve agent training.

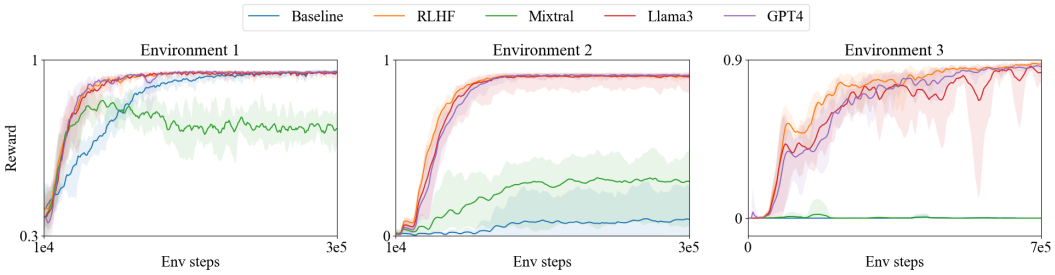

Figure 3: Policy training curves with RL-LLM-F rewards based on distinct LLMs

5.3 Reward-Function Scale Analysis

The magnitude of the reward signal is crucial for shaping effective reward functions for RL agents' learning. Large reward signals can boost learning with strong incentives, but would lead to detrimental outcomes if they are not representative of the task's objective. While small reward signals could potentially offer more stability and smaller variance, they might lack sufficient impact to drive the learning process. To further study the strengths and drawbacks of reward functions in RL-LLM-F through scaling, we deploy two scaling methods: (1) insert an output layer of `tanh` with a scaling factor to the state-score model, (2) post-process the calculated reward from the state-score model. The experiments are conducted with models trained from Llama3 data in Environment 2 and 3.

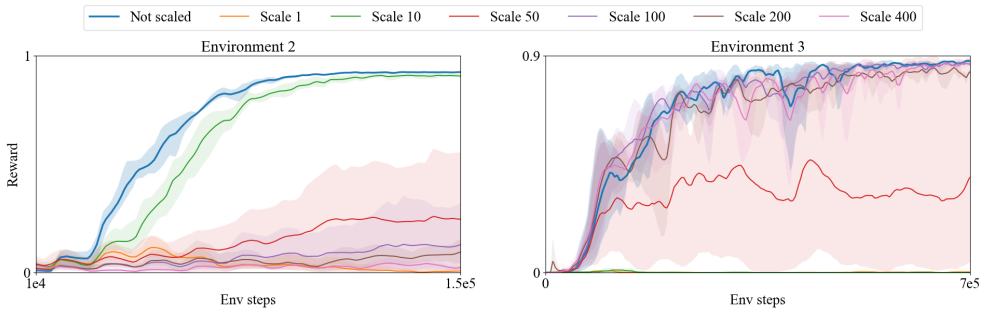

Figure 4: Policy training curves for state-score models with different scales using tanh.

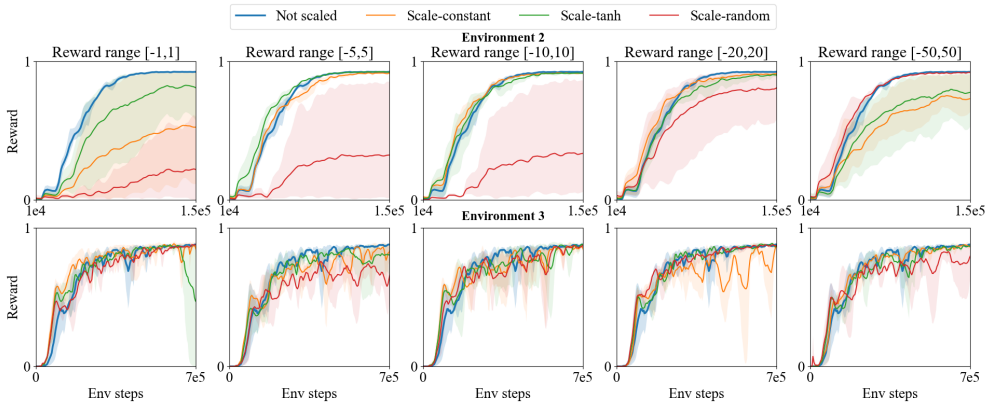

Figure 5: Policy training curves for rewards scaled to different ranges with different functions

The result of the first method is shown in Figure 4. For Environment 2, all 6 scaling factors performed worse than our RL-LLM-F without any scaling. While in Environment 3, large scaling factors can produce performance no better than RL-LLM-F. The possible reason is that the learnt state scores are naturally optimized to represent the preferences over the state space. The scaling method actually exerts range constraints on the state-score model, making additional difficulty to learning.

The result of the second method is shown in Figure 5. The scale-constant maps rewards to the min or max values of the range, the scale-tanh method maps to the values in the range, and the scale-random method multiplies rewards with a random scaling function $f(s) : \mathcal{S} \mapsto (0, 1)$. None of the post-processing tends to outperform the RL-LLM-F reward. Beyond this, the random scaling is shown to produce significantly worse average returns in 6 of the 10 experiment settings, while Scale-constant and Scale-tanh produce less bad results. These results emphasize the effectiveness of both the scale and shape of the reward function based on state-score models.

5.4 Impacts of Trade-Off Between Reward Density and Quality

Reward density and quality could affect the performance. The trade-off refers to whether it is beneficial to use a small number of rankings while they are very accurate, or lower the bar of accuracy to involve more ranking data that lead to a denser reward estimation. Empirically, LLMs perform badly with states related to rotation, so we exclude them from the experiments that have been presented so far. Specifically, the agent-direction information in state observation input to the state-score model is masked in previous experiments, so the state-score model will not give feedback when the agent takes turning actions, and the rewards of turning actions are replaced with a step penalty. This operation causes a decrease in reward density, but also an increase in reward quality,

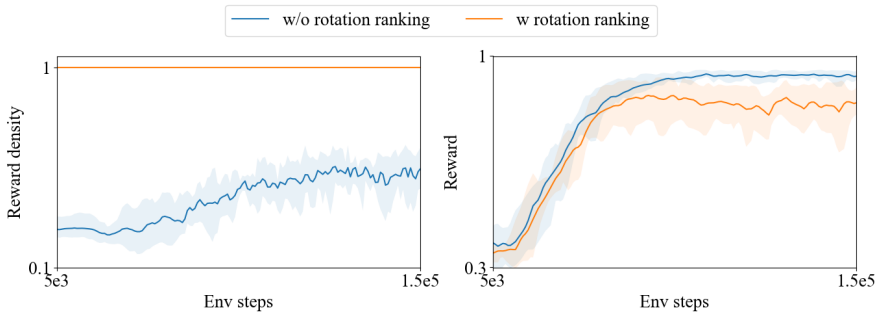

Figure 6: Policy training curves with and without rewards on turning actions in Environment 1.

considering the poor performance of LLM on ranking rotation. To further study the trade-off, we compare the performance where Llama3 estimates and does not estimate rotation ranking.

The result is shown in Figure 6. We use Llama3 in the first environment, where 2500 pairs are collected as training data for the state-score model without ranking rotation, but 6000 pairs with rotation ranking. Though it is more expensive, since Llama3 can only achieve an 88.3% ranking correctness rate in the 6000 training data, the ranking correctness rate of the state-score model can only achieve 84.4%, leading to much noisier rewards. In this way, though the reward density becomes more than twice as high as before, the policy training return cannot converge at a level comparable to that without rotation ranking. Therefore, it can be seen that the feedback quality plays a much more important role than the reward density in policy training with RL-LLM-F rewards.

5.5 Effectiveness of Rewards Based on State-Score Difference

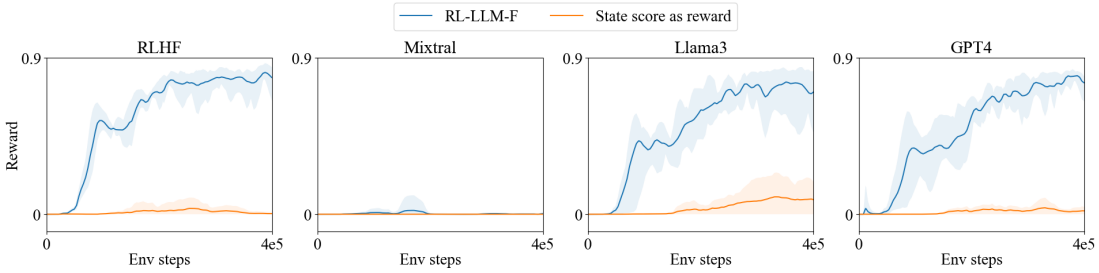

Figure 7: Policy training curves with state scores as reward functions in Environment 3.

To further prove the effectiveness of our reward functions derived from state, we compared our method with using state scores as rewards for state models trained with each LLM variant. The result is shown in Figure 7. The reward produced directly from state-score models failed to drive the agent to learn a good policy, as they have unreasonably large reward scales and might suffer more from reward hacking problems.

6 Conclusions and Future Work

We proposed a method to improve LLM-based RLHF, through estimating the rewards more accurately with a state-score model, and more efficient sampling. We plan to test on more complex environments. Moreover, we plan to handle the contradictory predictions from LLM to further improve the performance of our trained models based on LLM, with part of the results shown in Appendix A.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. *arXiv preprint arXiv:1910.04365*, 2019.

Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pp. 8657–8677. PMLR, 2023.

Layla El Asri, Bilal Piot, Matthieu Geist, Romain Laroche, and Olivier Pietquin. Score-based inverse reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, 2016.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021a.

Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*, 2021b.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. *Active preference-based learning of reward functions*. 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Andrea Lockerd Thomaz, Cynthia Breazeal, et al. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Aaai*, volume 6, pp. 1000–1005. Boston, MA, 2006.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A LLM-Ranking Improvement: Consistency Checking

LLM would produce self-contradictory answers to the same query sometimes, which is detrimental to the reward model training process that is based on maximizing the score difference between input state pairs. In our cases where state pairs are generated from a finite state space, a consistency check can be employed to improve the data quality. To be more specific, we remove or correct the contradictory state pairs. The scheme was tested on Llama3 and Mixtral, as the GPT4 ranks correctly.

For a consecutive state pair s_1 and s_2 with non-neutral preferences, we calculate the concordance index (C-index) c for the labeler's preference for this pair across all answers from bidirectional transition queries. c falls within the range between 0.5 (chaos and random) and 1.0 (no contradiction in all comparing results, harmonic), which could also infer the labeler's confidence in preference. Based on the C-index, 2 variations of data process schemes can be performed on the ith pair with C-index c_i compared with the C-index filter c_θ: remove the ith pair if $c_i < c_\theta$, and correct all less common preference if $c_i > c_\theta$. The former removes the pairs that LLM is unsure of, and the latter cleans up noise when LLM is certain. We compare choices of (1) Only Remove, (2) Only Correct, and (3) Remove and Correct. For all three choices, we tested with $c_\theta = 0.6$ and $c_\theta = 0.8$. For $c_\theta = 1$, only the first choice is tested, in which case no correction is needed for all c_i =1.0.

The result is shown in Figure 8. For the Llama3 variant, as we have seen from Figure 3, the agent trains to the optimal for the first two environments but has a big variance in the third (i.e. the "no consistency check" in Figure 8). We could empirically observe that the choice of (2) Only "Remove" with $c_\theta = 0.6$ converges with the smallest variance. This enables us to further improve the Llama3 variant successfully.

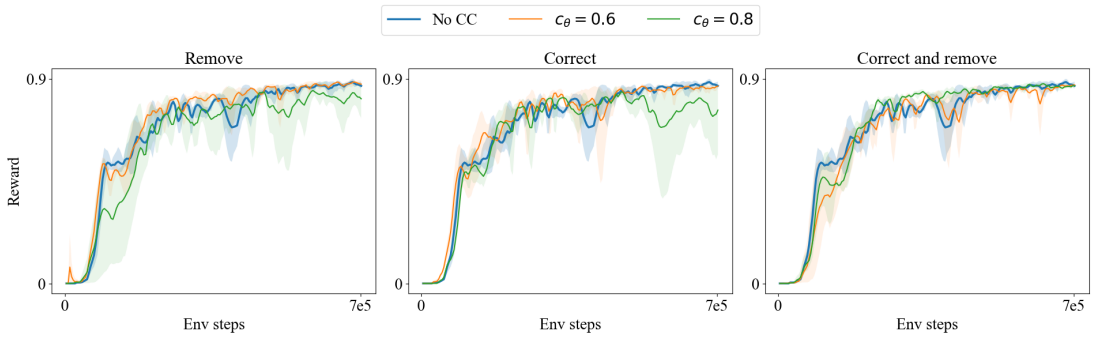

Figure 8: Consistency check results for Llama3 in Environment 3

B Example LLM Prompts

A typical LLM description prompt consists of three parts: environment and task description, example based on the Chain of Thoughts, and the question. Take a prompt for Environment 3 as an example.

Environment and Task Description

- Layout: The environment consists of an 11x7 grid divided into 6 chambers. Chamber1 occupies the top-left 3x3 section, Chamber2 the top-middle 3x3 section, Chamber3 the top-right 3x3 section, Chamber4 the bottom-right 3x3 section, Chamber5 the bottom-middle 3x3 section, and Chamber6 the bottom-left 3x3 section. A door at Chamber2 connects Chamber2 and Chamber3. Another door at Chamber4 connects Chamber4 and Chamber5. There is one Agent moving to a clinic in Chamber6 in this environment. The agent starts in Chamber1, and it must go to Chamber2 first, then to Chamber3 from Chamber2, then to Chamber4 from Chamber3, then to Chamber5 from Chamber4, and then go to Chamber6 from Chamber5. Every time the Agent can only move for one grid in one of four directions, up/down/left/right.

- Coordinate System: In this 11x7 grid, points are labeled (x, y), with (0, 0) at the bottom left and (11, 7) at the top right. Therefore, the coordinates span from (0,4) to (3,7) in Chamber1. The coordinates span from (4,4) to (7,7) in Chamber2. The coordinates span from (8,3) to (11,7) in Chamber3. The coordinates span from (7,0) to (11,3) in Chamber4. The coordinates span from (3,0) to (7,3) in Chamber5. The coordinates span from (0,0) to (3,3) in Chamber6.

Example

Q:
State[a]:
Agent: Chamber1 (2,4)
Passage to Chamber2: Chamber1 (3,5) right
Door at Chamber2 (7,5) to Chamber3: locked
Key in Chamber2: (5,6)
Passage down to Chamber4: Chamber4 (9,3)
Door at Chamber4 (7,1) to Chamber5: locked
Key in Chamber4: (10,2)
Passage to Chamber6: Chamber5 (3,1) left
Clinic: Chamber6 (1,1)
The agent does not carry any key. It needs a key.
Agent action: move up
Does the action taken by the Agent in State[a] help it progress toward the Clinic? Explain with Manhattan distance.

A: Let's think step by step.
First, what action should the Agent take to progress toward the Clinic in State[a]? Given the Agent in Chamber1, to reach the Clinic in Chamber6, the Agent must first try entering Chamber2. To enter Chamber2 from Chamber1, the Agent must first pass the passage at Chamber1 (3,5).
Then, did the Agent do so? The mahanttan distance between the Agent and the passage is |2-3|+|4-5|=1+1, which is two.
The Agent takes an action to "move up", which means it will move from (2, 4) to (2, 5).
Which chamber is (2,5) in?
Still in Chamber1. But the mahanttan distance between the Agent and the passage becomes |2-3|+|5-5|=1+0, which is one, so the Agent is indeed one step closer to this passage.
Therefore, the action taken by the Agent in State[a] indeed helps it progress toward the Clinic. The answer is Yes.

Question

State[b]:
Agent: Chamber1 (0,6)
Passage to Chamber2: Chamber1 (3,5) right
Door at Chamber2 (7,5) to Chamber3: locked
Key in Chamber2: (5,6)
Passage down to Chamber4: Chamber4 (9,3)
Door at Chamber4 (7,1) to Chamber5: locked
Key in Chamber4: (10,2)
Passage to Chamber6: Chamber5 (3,1) left
Clinic: Chamber6 (1,1)
The agent does not carry any key. It needs a key.
Agent action: move down

Does the action taken by the Agent in State[b] help it progress toward the Clinic? Explain with Manhattan distance.